Neues von IO::Socket::SSL

Steffen Ullrich
GeNUA mbH

Was ist IO::Socket::SSL

- Layer oberhalb von IO::Socket::INET
- Wrapper um Net::SSLeay (openssl)
- tied Filehandle

Was hat das mit GeNUA zu tun

- neue Release von GeNUGate kann in SSL Verbindungen reinschauen um auf Viren zu scannen, Protokoll zu forcieren...
- brauchten Bibliothek, die das für uns macht
- sollte möglichst analog zu IO::Socket sein, um die Änderungen in unserem Code gering zu halten
- IO::Socket::SSL erfüllte auf den ersten Blick als einziges diese Anforderungen

Wo war also das Problem

- benutzen eventgesteuerte, non-blocking I/O (viele Verbindungen in single Thread mit select-Loop)
- IO::Socket::SSL konnte das nur rudimentär, insb. nicht für accept, connect
- stellte sich heraus, das IO::Socket::SSL ein paar ernsthafte Bugs hatte, die seit langem bekannt, aber nicht gefixt wurden

Wie wurde es gelöst

- Kontaktieren des aktuellen Maintainers und Anbieten von Hilfe oder Übernahme der Maintainership
- Autor war offensichtlich froh einen Nachfolger gefunden zu haben
- Übernahme der Maintainership problemlos

Was ist neu in IO::Socket::SSL

- div Bugfixes
- non-blocking connect und accept
- Verhalten von non-blocking sysread, syswrite konformer zu Methoden aus IO::Socket, d.h. insb. bei Fehlschlagen wegen SSL Handshake gibt es EAGAIN
- Unterstützung für Zertifikatobjects statt Zertifikatfiles, sinnvoll bei on-the-fly erzeugten Zertifikaten für Man in the Middle
- Workarounds für existierenden Code

Unterschiede *::SSL zu *::INET

- SSL Handshakes laufen im Verborgenen ab, d.h. sysread muss eventuell schreiben und syswrite lesen
- wird angezeigt mittels `$! EAGAIN` und `$SSL_ERROR` auf `SSL_WANT_WRITE` bzw. `SSL_WANT_READ`
- Upgrade von *::INET zu *::SSL möglich via `start_SSL` (auch non-blocking)

Vorteile für GeNUA

- weniger Arbeit, da man auf etwas vorhandenem aufbauen konnte
- mehr Arbeit, damit die Abwärtskompatibilität gewährleistet blieb
- weniger Arbeit, da Nutzer außerhalb von GeNUA auch Bugs fanden
- höhere Zuverlässigkeit, da Module auch außerhalb der Firma viel genutzt wird (u.a. optional in LWP)

Vorteile für die Perl Community

- besseres Module, mehr Features, weniger Bugs
- Firma im Hintergrund, die das Module wartet, da sie es selber braucht
- Firma, die gesehen hat, das es Sinn mach auf diese Weise zu arbeiten, und es auch in Zukunft machen wird, z.B. mit Net::SIP

Offene Bugs

- Mysteriöses Problem auf Win32
- geht nicht mit Storable::fd_store,fd_retrieve, da dieses via perlio direkt mit sysread,syswrite auf dem Filehandle arbeitet und nicht die wrapper benutzt.

Ende